TS Bulletin *8-88

last of
/8 issues
last
year

For Low Cost Computing
published by
BILL HARMER
97 Ruskin St.,
Ottawa, Can. K1Y4B3

As the second year of publication of this newsletter draws to a close one can still see the Timex/Sinclair computers supported by small vendors and local user groups. All, in all, publishing this newsletter has been an interesting experience that, even if sometimes a hassle, has added something special to life for me. The future format and even the future itself of this newsletter is still subject to reassessment at the time of putting together each new issue altho it is hoped that some sort of newsletter publishing activity will continue from this desk.

This issue includes an article on the origin and early evolution of the microprocessors that have made home computing possible. An article on the programming language Pascal is also featured with this issue. Some coverage of the use and availability of disk drives for Timex-Sinclair computers, perhaps overdue, has been given based however on somewhat fragmentary information. Another article in our series on sorting (alphabetical and numeric) appears here too, under the title "It Takes All Sorts." All and all, another substantial read. Due to a small change in format, the articles are now longer, so that the equivalent of about one extra page is added to the new issues.

Further to our last issue's remarks on user groups, note that many user groups support several different computer makes and thus have Timex/Sinclair SIGs.

CONTENTS:

-Getting A Disk Drive For Your Timex-Sinclair p.7
-It Takes All Sorts (Alphabetic and Numeric Sorting). p.8
-Pascal On the Timex-Sinclair p.5
-Famous Microprocessors - The First Microcomputer CPU's p.3
-Famous Computers - The DEC PDP-11 . . . . p.4
-"What Do You Do With Your Home Computer?" p.6
-ZX-81/TS1000 Program Feature of the Month p.2

News:-Boston Computer Society,One Center Plaza, Boston MA02108 and San Diego Computer Society,c/o Tim Edwards,8155 Jamacha Rd., San Diego, CA92114,have active Timex-Sinclair SIGs(Special Interest Sub-Groups). Probably others too.

-Zebra Systems is selling ZX-81's in kit form these days. (About US$50).

-Timelinez which hadn't published for a while is back out. Addresses: Timelinez, c/o Geo. Mockridge, P.O.Box 1312, Pacifica, CA94044 and Andy Hradesky, Editor, Timelinez, c/o American Micro C.,2175 Aborn Rd.,Suite 262, San Jose, CA 95121. Timelinez is the newsletter of San Francisco area gps.

-*QL Emulator board for Atari ST has been reviewed by Oct."QL World"magazine.

-Two sources of TS2068 SCLD i.c.'s are E. Johnson(904-775-4935) and Paul Holmgren (317-291-6002)-Chicago TSUG newsletter reports (Nov.88).

-*TS1000 SCL/ULA chip is explained,Dec.SINCUS N/L,1229 Rhodes,Johnson City,NY

This is a public domain newsletter published in order to encourage the idea that you don't have to be well off to enjoy the home computing hobby and in general to promote the extension of computer literacy using low cost home computers like the Sinclair ZX-81, Timex TS1000 and TS2068. It is published as a hobby activity by Bill Harmer and sent to a few user groups supporting Timex-Sinclair computers. Back issues are still available (for about $2 each, to cover mailing and photocopy costs).* Warning: check with a second qualified and reliable source before you buy from or send money to anyone mentionned in this newsletter (addresses may also be out-of-date), hack or repair your computer, build any hardware project or make any comitments based on what is said or mentionned herein. And have fun computing above all!

© IN THE PUBLIC DOMAIN, 1988

Timex-Sinclair ZX-81/TS1000 Program Feature-Larken Disk Utilities

When you get your Larken disk drive system for the ZX-81/TS1000 you will want to convert already written BASIC programs to run on it and load from the disk. The programs in BASIC that you have written or keyed-in can be handled simply by calling the disk interface screen from the i/f ROM and keying in SAVE"MYPROG.B1"(for example). To make a self-starting program you might consider adding to the program the following module:-

```
9500 REM DISK SAVE
9510 CLS
9515 LET X=PI**PI
9520 PRINT USR VAL"14336"
9530 REM SAVE "PROG1.B1"
9540 SLOW
9550 PRINT "VERIFY?(Y OR N)"
9560 IF INKEY$<>"Y" AND INKEY$<>"" THEN GOTO 9610
9570 IF INKEY$="" THEN GOTO 9560
9580 LET X=PI**PI
9590 PRINT USR VAL"14336"
9600 REM DIRE
9610 CLS
9620 GOTO SGN PI
```

If you further enter say as lines 6 and 7 in your program LET DIRE=9580 and LET DISK=9500, then you can pull a directory using this access from BASIC (if the program with these lines has been RUN at least once and thus executed lines 6 and 7) by typing in GOTO DIRE in the immediate, command mode. You can also save the program on disk by typing GOTO DISK, in a similar manner.

Analysing the above program will show a few features such as the time wasting LET X=PI**PI which puts in a short delay to make sure that the disk drive is up to speed and you could also add line 9615 SLOW to ensure that the computer returns to SLOW mode since it goes into FAST mode on access to the commands of the disk interface screen. PRINT USR VAL"14336" saves memory over PRINT USR 14336 and GOTO SGN PI similarly saves memory over the equivalent GOTO 1.

Converting programs with a lot of machine language to run on the disk system is a little more complicated. Usually loading the program by tape with the command LOAD CHR$ USR 832"" (the latter-" -is shifted P), will cause the program on tape to load and then after a very brief pause go into the save routine. At that point between the two routines, the screen will momentarily clear and if you hit the BREAK key at exactly that time, you will be able to stop the program and it will be sitting in memory. Then you can LIST it with LIST 1, LIST 2, LIST 5 and so on and save it onto disk by using the immediate save: PRINT USR 14336 and press enter and then SAVE"PROGX.B1" and press enter. With a machine language program you may have trouble running it again due to copy protection features but this can often be figured out. Loading a commercial program from disk saves a lot of headache since machine language programs often crash while running, and reloading them by disk is a 7 sec. operation while a cassette reloading may take 2-7 minutes. The ZX-81/TS1000 is definitely more of a pleasure with a Larken disk drive system.

Famous Microprocessors - The First Microcomputer CPU's

The first microprocessor was the 4004 by Intel. It was designed originally for use as the processing unit in an electronic calculator on the order of a Japanese electronics manufacturer. It was decided that using ordinary chips would be too cumbersome and costly for the number of functions that a calculator would have. So the guts of a small computer patterned to some extent on the PDP-8 but with only 4-bit data processing capacity at a time was put on one LSI silicon chip. To make it work extra ROM and RAM chips were required, plus a few other functions. Of course someone had to sit down and write the machine language program in 4004 instructions to make it function as a calculator but when you are talking about mass production, such software is easy to duplicate into ROM's and much, much cheaper than adding extra chips for extra functions, even custom made chips. For a while the conservative sceptics called the microprocessor a useless invention (except for calculators)"looking for a use." Well, it quickly found them as we all know.

Intel then made an improved and expanded version of the 4004 which could process 8-bits at a time instead of 4. It was (not too illogically) named the 8008 and was seen as a programmable main chip for computer terminals that would improve on the dumb terminals of the time in having some local processing power. The formats of data being exchanged with the big computer to which it was attached in the data processing department could be modified by reprogramming the ROM used with the 8008. Such an intelligent or smart terminal could cut down on the processing time that a terminal would steal from the main computer, and computer time is definitely translatable into money.

When some amateur computer makers and experimenters saw the 8008 it occurred to them that a stand-alone home computer could be made from it for much less than the cost of a CPU built from transistors or smaller scale chips (like RTL or TTL chips). Now a computer could be made, particularly if sold in kit form to electronics buffs for home assembly, for a price that an individual could afford rather than a commercial or government lab or school. Soon Commodore Business Machines of Toronto thought that they could manufacture a cheap computer for the home, called in the manner of the 'pet rock', a Pet computer, to tap the market for experimenters who were not also electronic project builders. Now you did not need to be handy with a soldering iron to get a home computer. All you needed was about $1000.

The Commodore PET used the 6502, a different 8-bit microprocessor that was based more closely on the architectural economies of the PDP-8 computer, in that it had only one main register called the accumulator (while the 8008 had several). It was a further refinement of the economical design of the Motorola 6800 which had two main registers (Reg.A and Reg.B) and in fact was produced by defectors from the Motorola design team that produced the 6800, apparently believing that paring the registers down to two was not going far enough. Since the 6800 and 6502 had fewer registers they needed to read or write data from the main memory (RAM or ROM) more frequently. Since it takes longer to access this external memory rather than the registers on the chip this would normally slow down the speed of execution. The solution was to speed up memory access. The 6502 used an extra index register and both the 6800 and 6502 used a type of addressing called page-one addressing in which shorter and faster instructions in machine language can access memory in the first 256 bytes of memory. In addition, other methods of reducing the number of cycles to perform a given machine language instruction made the these microprocessors as fast or faster than the multi-register designs of Intel. The 6502 was used in the Commodore PET, VIC-20 and a variation on it, the 6510," used in the Commodore 64. The Atari 400, 800, 65XE and 130XE (the 8-bit Atari home computers) all used it. It was also chosen for the Apple II series, reputedly because the company gave away free sample kits to interest developers and priced the chip well below the price of the Z-80 which was the preferred chip even in the eyes of its designers at the time at Apple. (When Z-80 prices fell, they got a Z-80 board for the Apple for CP/M).

Meanwhile, Intel improved the 8008 and produced the 8080 and its very close cousin the 8085. 8085 improvements were mainly electronic rather than of computing significance, the 8085 requiring fewer power sources than the 8080. The 8080 became the standard for the operating system of the time, CP/M, the first microprocessor standard in disk operating systems and predecessor of them all, MS DOS, OS/2 and all other microcomputer originated disk operating system (Universal standards). The Z-80 would handle programs written for the 8080 as well as adding a few features made possible by the use of new VLSI chip technology for getting more on a chip. The Z-80 soon became the new standard used in Timex/Sinclair, TRS-80 and many other microcomputers. Thus started an era which may topple the giant computers yet!

Famous Computers: DEC PDP-11 Minicomputer

First released in 1970 about 300,000 of these minicomputers have been sold in various models. The early ones used a CPU made of separate largely TTL chips and the latest model, the microcomputer version uses an F-11 microprocessor on one chip. The RAM varies from 32K to the megabyte range with 256K being a standard, recent version complement. Early models used an 8" floppy but later modles used a 5¼", single sided floppy (or usually two) with 400K each. A 10Mbyte hard drive is also available. Although largely made obsolete for big installations by the VAX, the PDP-11 was well enough thought of to issue a microprocessor based version in 1983 using the same (IBM-style, floor mounting) case as the Micro-VAX. And indeed when early versions of the VAX 11/780 came out they had a PDP-11 compatibility mode so that they could run PDP-11 software without modification.

The PDP-11 can use a number of operating systems supplied by VAX including VENIX, a UNIX-clone but the most popular over the long run has been RT-11 (R for Real, T for Time). This operating system allows dual use of the minicomputer rather than true multitasking, so that the computer can be controlling an industrial process or scientific experiment while at the same time the operator can run little data processing or calculating jobs or work on editing and testing a new program.

The computer divides its memory into 8K pages (of 16-bit words) which are mapped into a virtual memory space (set of addresses) of 4 megabytes. Thus the hardware and in specific the Active Page Register(made up of a page address register and a page descriptor register) maps the actual pages of existing RAM into a 4 megabyte (virtual) memory map. Although only say 64K of actual RAM might be available (on a 64K machine), the 8K pages are assigned to areas of the vitual memory map in such a way that when something not in RAM is going to be needed in say the next block of virtual memory, an unneeded page is overwritten with 8K of program or data from disk and that block of RAM is assigned the virtual memory addresses that the computer will access when needed next. This virtual memory hardware which very few microprocessors have is really the best way to handle multi-user, multitasking, multi-programming applications and was a feature of the standard-setting IBM 360 introduced in the early 1960's.

The PDP-11 memory maps the peripheral input or output devices into the top 8K of memory much like the 6800 and 6502 and unlike the 8080, Z-80 and 8088 families which use separate input and output commands that address I/O ports (256 possible ones in the Z-80). The Micro PDP-11 sells for about $10,000 plus dollars although accessories are expensive for it and very few games are available for it (the exception being a machine language zap-em game written as a project by a local Algonquin College technology student a few years ago and brought out on open house nights). So it would not be a very practical computer for the amateur to own (although old used models often come cheap). A teletype terminal or VT-100 terminal can be used for operation (no built-in keyboard like most home computers!) and a typical DEC line printer costs about $9,000 new.

Although the PDP-8 architecture probably influenced more microprocessor designers than any other with its(component economical) single user register (accumulator) design philosophy, the PDP-11 follows more the multi-register idea having eight user accessible registers, R0 to R5 being general purpose registers and R6 the stack pointer and R7 the program counter (to say where the CPU would find the next instruction to execute). Register-to-register and too, register/RAM operations are available as well as a wide variety of somewhat more complex memory access modes in each command. The full complement of address modes are allowed for all the general registers (R0-R5) so that a very rich instruction set indeed is available. In this it resembles the IBM 360 rather than the PDP-8. The registers are 16-bits wide and memory is usually addressed in 16-bit chunks (or words as it is better called) although it can also be addressed in and operated on in bytes (8-bits). In addition to machine language programs (seen in the disk files as xxx.MAC) and BASIC programs (xxx.BAS) , FORTRAN, COBOL, Pascal and C language are available depending on the operating system used. As with the PC-compatible computers, compilers usually only operate with one operating system so that the BASIC compilers for the CP/M version for the IBM compatible computers work with that CP/M86 operating system, Quick BASIC or TurboBASIC work with the MS DOS (or PC DOS) operating system and will only work with the new OS/2 operating system when it is running in MS DOS mode (a feature given it so that it would not obsolete the large store of programs written for MS DOS). The PDP-11 compiler for C language is for example only available to run under the RSX-11S operating system, not the more common RT-11.

A generation of college students have learned assembly language programming using the PDP-11 (although today's students have been called the VAX generation) and it still soldiers on in many a factory, lab and school although largely considered to be obsolete. It is a computer that many have an affection for and which has influenced the thinking of many designers and programmers. Not as useful as a Commodore 64 for the gamesperson and available in various old models from huge to merely space consuming sizes (in the non-microcomputer models), having memory from magnetic core storage to silicon chip ic's, this computer spanned the middle generations in its 18-year life (to date). The VAX may have replaced it in many colleges, the microcomputer in many schools but a certain generation of technical people have memories of cutting their teeth on it as programmers and for them it will always live.

Pascal On The Home Computer

The first thing that one notices about Pascal, the programming language that is a competitor of BASIC is that it is not spelt all in capital letters. This is because the letters are not taken from the beginnings of other words as BASIC comes from Beginners' All-purpose Symbolic Instruction Code but is a person's last name, that of Blaise Pascal (lived 1632-1662) who invented one of the first calculators as well as the mathematical field of probablility. The inventor of Pascal, the computer language one Prof. Nicholas Wirth of Switzerland wanted a language which was more organized and systematic (dare we say structured, the word that is usually used of it) and chose the name of a historical figure known for rationalism to convey this association (at least to those who have studied intellectual history which means Europeans, I guess).

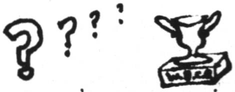
How is it structured? BASIC which took its form from FORTRAN in this allows the programmer to define a name made up of a bunch of letters as a variable which can be used as a number in formulas and moreover defined at any point in the program. This makes it easier for programmers, who do not have to plan ahead, and can just define a variable anytime they need a new one. This slows down the computer however. In Pascal you define the variables at the beginning of the program module. This suits the computer better as the variable tables (variable names matched with their values numerically) can be set up all at once rather than continually being added to during the running of the program as with BASIC. This makes the programs run faster and more smoothly. If you don't remember to add enough variables at the beginning you have to redo this part of the program. Some programmers get around this by defining a few extra variables at the start just in case they are needed and then deleting them in the final version of the program if they are not needed.

There are a number of vocabulary differences in Pascal that the new user simply can memorize and be done with. Instead of LET X=5*A in BASIC, Pascal would say X:=5*A and instead of PRINT one says WRITE and uses a little different format. There is also a variation on WRITE, WRITELN (write-a-line) that is also used instead of PRINT. IF...THEN...ELSE is similar to the IF...THEN...ELSE in BASIC except that a whole program typically follows the THEN (& ELSE)(even if it has only one line) rather than the BASIC which is typically followed by only one line/statement which can however call a whole program routine or subroutine if it is a GOTO or GOSUB (or in some BASIC dialects just a line number to GOTO). READ(X) would be used instead of INPUT X of BASIC. There is also a whole system of stating where a sub-program or sequence of instructions begins and ends. Typically BEGIN and END are used as markers,&the semi-colon (;), and indentation so that parts of a loop are lined up with the left margin in correspondence with their order of priority in the loop. The indentation particularly takes a little getting used to. The shortest way to mastering it is probably to read and analyse other's programs. Additionally, the operators DIV and MOD used with integer compilers instead of / (divide) might throw the new user coming from a floating point BASIC but the theory of use of these is beyond the scope of this short article. The one remaining peculiarity of Pascal is the use of lower case letters rather than CAPITALS for variables (of course only in computers that allow lower case letters at all).

Once you take the trouble to try to master Pascal you will be doing the sort of computer programming that is current with big computers (where BASIC is seldom used) and favoured by university computer departments. Pascal programs also typically run faster than BASIC programs that do the same thing. The so-called structured nature of Pascal also forces you to do a little thinking and planning ahead which is useful. The old adage says that the sooner the program writing begins on a big program the longer the job will take. The theory behind that is that the sooner you begin writing program lines the less time you spend on thinking the programming problem through and planning the program flow so that pitfalls and complications are forseen in advance and thus worked around from the start rather than fixed as you go on. As you well know, every time you fix part of a program that affects other parts of the program that must then be changed to work with the new part, which affects other parts that work with those parts...well, the time and extra work you can see, just pile up and compound themselves like the interest on the Indian claims to Manhattan Island hundreds of years ago.

All in all, if you are getting bored with your computer why not try something completely different like Pascal. There is Partial Pascal for the ZX-81 and 2068, and HiSoft Pascal for the TS2068's equipped with a Spectrum ROM.

"What Do You Actually Do With Your Home Computer?"



There is no perfect and therefore no one way to approach programming, and this goes for microcomputing as well as the big boys. In fact the home computerist that is into writing programs will want to take a different approach than the big boys for several reasons: he has only himself to please ultimately, the whole purpose is to tailor the finished program to satisfy the unique need that is the goal of the program whether it is a bright idea seemingly from out of thin air or a practical use of the computer, and time is not critical but a flexible commodity. Neither is foolproofing the program the first concern (because ultimately the programmer is the most important user). This means in practical terms that the average home-grown program is of medium length (5-15K), simple in conception (if complications arise they are added in increments as the programming goes forward), and represents some sort of challenge, learning or just "mucking around" satisfaction for the home programmer.

Where do the ideas come from for originating one's own, unique program? The possible sources for ideas are as diverse as the backgrounds of the home computer users. From hobbies, to work, sideline businesses to church or club mailing lists or files, science and technology to craft and pure math, fun or frustration of using a poorer program or not having the access to a computer which runs a particular commercial program. When the local college that used PDP-11 minicomputers to teach electronic technicians computer science found a class of 15 competing for time on two computers with 60 full-time students, the writer got the idea of simulating some of the assembly language instructions on the Timex-Sinclair TS1000 (in BASIC) in order to have something to practice on other than a $20,000 purchase of a real PDP-11. Seeing professional programmers botch the job of simulating the PDP-11 on a Cyber mainframe at the college, only made the job more enticing.

Some people potter around with pottery, some are exposed to photography as a hobby, some like to watch the weather. The result:- Timex-Sinclair programs to mix pottery glazes, predict the weather with barometer readings and make databases of slides and negatives. Some like to talk, even with a computer. The result is work on different versions of Eliza, the simulated conversationalist program. One young programmer even adapted it into a version he called LeRoy to simulate the banter of Richard Pryor (on the Radio Shack Color Computer). The writer too has had a go at Eliza-type programs, and did a demo of one (as far as it was completed) at a club meeting. It even would detect nonsense words (by checking to see if there were more than three consecutive consonants in a word) at which point the computer would rebuke the offender for talking nonsense to it!

Disk drive systems are another fertile field. There are never enough utilities with any but the largest system, for disk drive users. The result is the effort to write new utilities, to sort directory entries into alphabetical order, to save the directories of a number of disks (in the array variables of a BASIC program for example), to compress text programs, and to do anything that any other disk system has ever done that might strike the programmer's fancy and want his disk system to do too. Likewise, programs to adapt printers to work with your computer, print tall or lengthwise letters, small or user specified characters (such as Greek letters), to print pictures or graphic designs, on the screen or the printer, all are grist for the amateur programmer's idea mill.

The computer hardware hacker will find as the writer has, an unlimited potential for computer program creation in the electronic design and analysis field. Many ready-made programs can be taken from books or magazine articles and transferred to the BASIC dialect of one's own computer. Procedures for solving problems given in textbooks and electronics courses can readily be computerized by those used to doing them with pencil and paper. A computer can once it is programmed to do a given calculation at the expenditure of 4 or 5 hours of time, then do the calculations given set after set of different circuit values, in minutes ever thereafter. Too most of that time is that taken to enter the new parameters, not actually the computer's time taken to do the calculations and display the answers.

So looking over the selection of programs in various newsletters, it is safe to say that no one can hardly imagine what sort of programs exist for a widely held home computer of a given make. In fact the new workhorses of the programming field will be those user groups that make a country or continent-wide library of programs available for a given make and model of computer. Let's support them! -

THIS MUCH DISCUSSED QUESTION WINS THE... TROPHY for..... {WORLD'S BIGGEST COMPUTER QUESTION} ...YET AGAIN THIS YEAR!

GETTING A DISK DRIVE FOR YOUR TIMEX-SINCLAIR

I am sold on disk drives. The Larken system for my TS1000 has made it quick to work with indeed instead of the plodding mind-tool it used to be. Saving (or loading) a program now takes 7 to 10 seconds instead of the one to ten minutes that the cassette recording used to take. It stores about 158K to a disk costing 60-70¢ in Canada (5¼" double sided). This is 16 average programs. I am a believer.

Let me apologize in advance for the possibly incomplete information below. But a survey, even if it contains the odd mistake, is necessary to put the availability of disk drive systems into a larger perspective. So, here goes!

For Your TS2068: Aerco (Texas) makes a large and expensive system but it has many extra, non-disk related features such as peripheral interfaces and extra system RAM expansion. Also RP/M, a CP/M operating sys. clone is available as an option.

Larken Electronics has produced three generations of disk drive systems, LDOS & cartridge-port DOS among them. Fast, flexible and reliable as well as fairly inexpensive (as these systems go).

Oliger Co. has a disk system out too. Like the Larken system it will handle a variety of disk types, single or double sided, if you want it tailored for one of these (by special order).

RAM DISK-Larken El. also has a RAM disk which is used with the cartridge DOS interface to allow saves and loads to static RAM instead of the mechanical disk drive.

For The TS2068 Running the Spectrum ROM Using a ROM Switch

Larken systems are compatible with both TS2068 mode and Spectrum mode. Check with the other manufacturers to see if their systems have been adapted to handle Spectrum mode.

Disciple interface from England will work on the TS2068 only when it is in Spectrum mode. It has a snapshot mode to save to disk copy protected Spectrum programs from England.

TS1000/ZX-81: Aerco also makes a ZX-81/TS1000 version.A CP/M clone, RP/M is not yet available for it last I heard. Priciest but kits available too.*

Larken also makes an LDOS interface for the TS1000/ZX-81. Standard mode uses 5¼" double-sided diskettes, 158K approx. It is cheapest.

Oliger also makes a version for the ZX-81/TS1000.

Sinclair QL: Trump Card and the cheaper ACE Card from Miracle Systems are tops.

Sandy Q-Board (without RAM chips) is the next down in price.

Cumana Interface costs the least of the three (at~US$130).

Sir Clive's Z-88 Laptop has the ability to write to the disk drive in an IBM-compatible computer as well as storing information in its own ROM chip cartridges. It can write to disk with QL and other computers.

???? -PC8300 (oriental ZX-81 clone) and TS1500 - Not known if disks are available that will work on these systems, or for ZX-80.

Other disk systems were made for these computers and some individuals have made their own disk systems over the years. You might therefore find a non-standard system available on the used market or plans for making a do-it-yourself i/f.

Most disk systems' disks are incompatible with each other. Larken ZX-81 disks won't work on the TS2068, although with a switch of ROM the LDOS disk interface (the original one) will work with either the ZX-81 or TS2068. Often disk drives (from manufacturers of generic disk drives rather than whole units with interface as used by the Commodore 64) can be used with several different interface boards and/or for several different computers. If you now own a disk drive or can buy one used or surplus you may be able to find one of the above interface boards will work with it (often only specifying a special ROM is necessary).

Reviews of an early, prototype version of the Larken TS2068 LDOS system as well as the Oliger system have been printed in previous editions of the Ottawa-Hull TSUG newsletter. They were both positive reviews. Members of OHTSUG have used the Oliger TS2068, Larken TS2068, Larken TS1000, & Disciple (UK) (for Spectrum), disk drive systems so that a fair amount of experience has accumulated. They all agree that having a disk drive system is terrific.

* Through special deals with one TSUG.

It Takes All Sorts or Getting Your Act In Order

ABC.... 1234....

Way back, in ye primordial data processing times, punched card era, all data was on a deck of punched cards. IBM cards were the most famous, 80 columns each, one number (0-9) or letter (A-Z) per column, rectangular holes as if made by a builder's spike, well, you get the picture. Then you shuffled the decks into order according to some of the data and when they were in the right order for making a thick report (hardcopy listing) you took them from the card sorter to the accounting machine which would print out everything on the card plus cumulative totals or just part of what was on the card. To sort in order of last name, if 10 columns were allowed for the name, you had to put the cards through the sorter at least 10 times (once for every letter) and gather them up from the pockets (0-9) and restack them in that order. Oversimplified a little but you get the picture.

Then came computers. They also used punched cards. But with their limited capacity sorting was a problem as everything would not fit into the computer's tiny memory. So sorting went very slowly. Soon, memory got cheaper and this was no longer a problem. With home computers, everything got much cheaper, and no database program worth its salt, would be without a sorting routine.

The most primitive sort is the bubble sort. This was mentioned in detail in our newsletter #11-87. It goes through the list of items and takes each pair of adjacent (next to each other) entries, tests to see if they are in alphabetical (or numerical order if they are numbers) and if they are not switches them. Thus for this pair of entries, if they are not in order, the last shall be first and the first last once the routine has gone by this pair. A couple of nested FOR...NEXT loops and the alphabetical comparisson provided in BASIC, IF A$ < B$ THEN GOTO... are sufficient as far as tools are concerned. To sort in reverse order change the direction of the comparisson (e.g. from A$<B$ to A$>B$). Then your data will be listed starting with Z and ending with A or if they are exam marks, they will start with the highest marks (of the brighter students and bury the ignominy of the dull ones deep into the bottom part of the page where they probably hope no one will notice). The whole data base may be sorted even if the data is say the name in array N$(X) and the mark in array M(X) and the home room in array R$(X), if while comparing M(X) and M(X+1), you switch both the M(X) elements and the R$(X) and the N$(X) too, when switching the order of the pair is called for. See our newsletter *2-88 for a fuller explanation of that and a sample program in ZX-81/TS1000 BASIC. Since as far as the writer knows there is no patent protection on the bubble sort (or most others?), anyone can use it in a program of their own writing. It is found in so many different books for so many different home computers that there should be little problem finding one to run on your computer no matter what the brand or one or another version adapted to your computer's dialect of BASIC. Any way it is fun to write a simple database and experiment with sorting routines or just enter one from a book. For example Tim Hartnell's Getting Acquainted With Your ZX-81 has a 'Data Sort' program for the ZX-81/TS1000 to sort a list of numbers that is probably the shortest and simplest of such programs to enter into your computer.

Since that many other sorting routines have been developed. Some IBM-compatible BASIC compilers seem to have been optimized for certain sorting routines supplied with them. In other cases the programmer should choose the routine from a bunch of them according to the application. For example, the time that a given sorting routine takes to do its work depends not only on the number of entries but also on such things as whether the list is completely in order, only has a few items out of place (as when you might have just added a couple of new members to a membership list that was already sorted), or more or less random. Some sorting routines take longest in sorting an already almost perfectly sorted list. And of course sorting a list that is already in order is a waste of time if this is frequently the case, so that a routine to check if the list is in order before sorting might be good to call in first. Some routines to solve this problem include the flagged bubble sort (see for example, The Essential Guide to Timex/Sinclair Computers by Morse et al., Simon and Shuster, Pub.).

When you want to have the option of sorting a list of students and marks by student name or by the mark itself you could use another sorting routine seen in the last mentionned book rather than having two separate and complete sorting routines (such as bubble sorts), one for sorting by mark and the other for sorting by student name. There are Shell Sorts, Quick Sorts, Insertion Sorts as well as the Bubble. The fun comes in in trying different ones and timing the results. Go ahead. It takes all sorts....